ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laurea Magistrale in Ingegneria e Scienze Informatiche - Sede di Cesena

MULTI-AGENT SYSTEMS FOR NETWORK MANAGEMENT: STATE OF THE ART

Corso: Sistemi Distribuiti LM

Tesina di:
MICHELE ROBERTI

Prof:
ANDREA OMICINI

ANNO ACCADEMICO 2015-2016

Indice

Capitolo 1

Network Management

1.1 Network Management and relative Issues

Network Management systems have to evolve quickly to satisfy the customer's requirements. Reading the proceedings of international symposium on Network Management, it is clear that there is a strong need to go to flexible and scaleable platforms to fulfill the market expectations. Intelligent Agent paradigm as it was stated in the introduction seems to be a solution to these problems, but if most all Intelligent Agents use networks, only few are used for Network Management. Only two per cent of the official Agent product and research activities referenced by the Agent Organization are related to Network Management.
Why? We see at least two reasons for that:

- Firstly NM is still strongly influenced by OOP, and the huge development effort brought by the companies developing network solutions are always on the way. Most of them focus their strength to use CORBA which is a logical extension of their Object Orientation.

- Secondly, if Agent Oriented Programming has a strong potential in all computer sciences, like the Object technology 10 years ago, it represents a big cut with the classical development techniques, and is not yet mature.

1.2 Network Management and ISO

Network management has been and is always studied by International Standard Organization which defines an architecture and a set of standards. These standards are divided in three main parts:

- Management Communication (CMIS/CMIP) (ISO/IEC 9595 ...)
- Systern Management Functions (ISO/IEC 10164-*)
- Management Information (ISO/IEC 10165-*,10589, ...)

The functional side of the OSI Management is split in five strongly linked areas. Functional units are defined and used to cover the functional areas.

- Configuration Management: startup, shutdown, setting, reading, modifying configuration
- Fault Management: detection, trace, location, analyze, prediction, correction
- Performance Management: measurement, data collection, analyze, tuning, testing
- Security Management: controlling access, transfer , testing
- Accounting Management: set quota, verification, planing, billing

The OSI defines how a "managing system" may communicate and act directly on a managed system which may be either another managing system in charge of a part of the network, or directly an agent. The management is viewed through the five areas of management (FCAPS), which use "management functions" to perform the management activities. These functions use services management protocols (CMISE/CMIP) to access the managed object within the Managed System. The managed system is represented by a management Agent, CMIP Agent, which is in turn in charge of performing on the objects what has been requested by the managed system, or to send event reports when necessary. In comparison to the Internet standard of network management which is SNMP, OSI management standard are complex and costly to implement.

1.3 Internet Network Management

The Internet Network Management is a flat management based on protocols of communication between managers and agents to access device Management Information Base (MIB) which identify the variables that can be managed at the agent level. This simple and direct access to devices to be monitored and controlled explains why there is no need for special standards describing the Internet Management Architecture. But the other side of the coin is that SNMP is not usable to manage wide network because of, intensive use of polling which consume lot of bandwidth, problem of packets size to get back MIB data, lack of security, impossibility to have intermediary network manager (manager to manager). Therefore IETF has been obliged to propose improved version of SNMP to solve these problems. The main extensions of the far more complex SNMPv2 are:

- Protocol: new primitives GetBulkRequest and Inform Request
- SMI extensions: new data types, table categories
- Manager to Manager capability: Inform Request, Get Response

The second improvement proposed by IETF is RMON specification developed to perform proactive LAN monitoring on local or remote segments. Both SNMPv2 and RMON2 are a step towards distributed management, but the principle is still to have a centralized management platform. Even if there is no architecture provided, and only low level operations, the Internet NM is far away the most used because of its simplicity.

1.4 Intelligent Agents and Network Management

Therefore it is definitively necessary to review the way to design Network Management Architecture, with the mind that we will have to use intelligent entities to do the work. Some people propose new ways to develop Network Management Architectures, as Aiko Pra [1] who presents kind of prototyping approach of Network design, or Stevenson [2] who starts from the "help Desk point of view" to gather the most important NM Requirements and who emphasis that Artificial Intelligence needs to be applied

at all management levels. Studying the great potentiality of the Agents, we feel closer to this last approach, thinking that we have people able to perform activities, or who may acquire skills to perform them, and we will have to organize the NM activities around them. The Network Management Architecture becomes then a problem of organization, where are taken in account the problems encountered by the field services, or help desk. Several studies are done around what is called the Management Policy, which is based on the Object Oriented paradigm, but prototypes that have been developed are too much influenced by OOP and seem restrictive compared to what may be done with Intelligent Agents. The interest of these studies is the possibility to provide an architectured approach of Network Management with Intelligent Agents, keeping the simplicity supplied by the Internet management protocols.

Capitolo 2

Intelligent Agent Principles

2.1 Agent properties

Know what properties Agents may own is useful to understand their behaviour and how the will change our life in the future. Below are described the main properties:

Autonomy

An Actor decides himself when and under which condition he will perform what actions. The autonomy is explicitly required as property, but also the reactive, pro-active and social behavior. *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the further"* [3]. M. Wooldrige disagrees with Franklin and Graesser's definition of agent, and in "A response to Franklin and Graesser" the autonomy is explicitly required as property, but also the reactive, pro-active and social behavior. One question that we have longingly debated is "what is the autonomy without pro-active or reactive or deliberative behavior?". Someone gives viruses and "networked viruses" also called worms as example of autonomous, and mobile agents.

Communication

Communication is one of the key property of agents; it is the ability to speak with a peer, with human (Interface Agents), or with devices.

Collaboration/Cooperation

Agents are collaboratives when they are able to work together. Also the agent is able to communicate and negotiate with the others, he is delibera-

tive and may coordinate his actions with the others. Collaborative agents are usefull among other things when a task involves several systems on the network. The negotiation is the main issue of the collaborative agents. If coordination may occur without collaboration, collaboration needs negotiation.

Deliberation

Know rules, and apply them without waiting for instructions. M. Wooldrige and N. Jennings [4] define an deliberative agent *"as one that contains an explicitly represented, symbolic model of the world, and in which decisions are made via logical reasoning, based on pattern matching and symbolic manipulation"*.

Mobility

Since Java had appeared, we may find a lot of mobile Agents, but there is different kinds of mobility:
- The mobility which allows the agent to move from one system to one similar one
- The mobility which allows the agent to move to another different system
- The mobility which allows the agents to suspend their action on one system, move to another and go on.
- The mobility which allow the agent to move himself, versus to be transported.
- The mobility which is a duplication of the agent to another system (cloning).
- The agent carries his knowledge to another system.

Learning

There is a least two definitions of learning: The bad one: an agent is said to learn if he is able to acquire knowledge (data). The good one: an agent is said to learn if he is able to use is new knowledge to modify his behavior. Despite the fact that learning is an important factor of intelligence, there is only few agents able to learn. Most of the time they have fixed (precompiled) rules and knowledge base. The objective of learning is for the agent to perform new tasks dynamically without being stopped.

Pro-activeness

"Pro-active actions are intended to cause changes, rather than just reacting to change". Pro-active agents generally follow plans, or at least execute rules when the environment reaches a known threshold. Sometimes proactive is used with the same meaning as deliberative, but an agent may be

pro-active, because he has been requested to perform pro-active tasks, as opposed to deliberative agents, who decide themselves to be pro-active.

Reactivity

Do something when an event occurs.

Security

Is able to discriminate friend from enemy and contaminated elements.

Planning

The agent organizes by priorities the actions to perform during is life. For many researchers planning is one of the most important properties for an intelligent agent to possess. Planning is used by deliberative and pro-active agents according to their knowledge of the environment and the possible actions to apply to it.

2.2 Agents Technologies

The development of agents requires languages to build the agents, protocols and languages to give software the possibility to communicate, and languages to describe and transfer agent's Knowledge, Beliefs, Goals, Desires, Intentions [5]. The main idea for our purpose of Network Management, is the possibility for the agent to exist on different heterogeneous environments. Thus they need a standard infrastructure on each system where they need to be hosted. Then agents may be developed like if they will be always on the same machine called the Virtual Machine (VM). The easiest languages to create an "agent" are script languages that may be then executed on remote system, because they are interpreted as shell languages are on Unix systems. The most known are Telescript from General Magie, Perl and TCL (Tool Command Language) from Sun, and of course Java which is an Object Oriented scripting Language. The second class of languages contains languages coming from classical programming approach as LALO (Agent Oriented Language) which generates C++ sources, or AI oriented languages as Obliq from DEC. Depending the languages and tools used to create them, Agents are born with more or less usable "Intellectual capabilities".

As regarding the knowledge, it is described using another layer of language, that we may compare with our respective languages French, English, Portuguese, German, ect. One usually used knowledge language is

KIF (Knowledge Interchange Format), but specific languages are usually developed within projects for specific purposes (e.g. GDL for MANIA). It is sometimes not well perceived that Intelligent Agent paradigm use a human view and speech to present what is only software. This approach is nevertheless normal and accurate, because we are at the stage to use Intelligent Agents as if they were human workers, given them responsibilities and autonomy in their actions. In the next chapters are presented the experimentations using agents to solve the problems encountered by the classical Network Management approaches:

- Scalability: when systems are growing or new services are added, Agents may take in charge the new environment, by changing theirs roles, believes, organization.
- Robustness: the fully distributed versus centralized Network Management is more robust because NM operations may continue even if part of the network is disconnected from the Master management site.
- Upgradability: new Agent may replace oldest without stopping their activities.
- Performances: closer from the systems to manage, the distributed management allow faster reactivity and even pro-activity, and less resources consumption.

Capitolo 3

Agents through Management by Delegation

People working in the NM field have quickly identified deficiencies in classical NM architectures and protocols. Management data is distributed among agents with very limited processing capabilities, while data processing to extract management information is centralized in managers. The latter consequently suffer from heavy processing load. Furthermore, management operations must be decomposed to very primitive functions mostly limited to SET- and GET-like primitives and then communicated to the agents. This is present in micro-management [6]. Micro-management leads to heavy management traffic in the network. To overcome this uncomfortable situation, and in order to define a more flexible and scalable architecture with real-time management capabilities, it is tacitly admitted that management functions and operations should be dynamically carried out close to where the managed objects are. Management by delegation is designed following this principle.

The main concept on which MbD relies is the elastic server [7]. An elastic server is an enhanced server whose functionality can be extended and contracted dynamically during run-time. A delegation protocol allows a client to pass new functions to an elastic server and then asks the server to execute these functions by instantiating them. It also allows to keep control on each instance so as it can be stopped then resumed for example. A new functionality is prescribed in a script with no restrictions on the language in which it is written. Applied to network management, elastic servers can be

interpolated between managers and agents (in a similar way to middle level managers in organizations) with similar roles to proxy agents. These elastic severs are then called MbD agents. An alternative approach would be to make classical SNMP /CMIP agents evolve towards elastic agents. But this approach is rather rejected. Indeed, manufacturers usually tend to keep the legacy systems as they are and to design ad hoc components instead. The manager uses the delegation protocol to upload management scripts to the flexible agents. Using the same protocol, it asks to instantiate some delegated script and to run it. Therefore, the management operation prescribed in this script can execute "autonomously" in the same environment as the managed objects. The manager is able to control its execution via dedicated primitives in the delegation protocol. Later, Goldszmidt [8] called these scripts "delegated agents". Under the banner of intelligent agents, some researches are being conducted to explore the MbD paradigm for NM purposes. In that, suggests a spreadsheet scripting environment for SNMP. The spreadsheet scripting language allows a manager to prescribe computations that can be carried out by the agent. Each cell in the spreadsheet is defined by an expression that computes a value from other given data such as values in the MIB. Expressions can be inserted, updated and deleted according to the manager needs. Each agent contains functional objects with network management functions allowing to access the management objects and to communicate with other agents. Therefore, a manager needs only to delegate script skeletons invoking these management functions. When running an instance, the management functions' invocation are bounded to those implemented in the functional objects. A same management function could be implemented differently on different agents according to the network device specifics for example. Alternatively, it's possible to add a rule processing unit to agents. Rules can be transmitted from managers to agents, which can endow the latter with intelligent behavior. Rules can be dynamically loaded, updated and removed, making the agents flexible.

3.1 Recapping

The implementations of the Management by Delegation concepts presented above are far from the agency's concept though they are presented under the banner of intelligent agents. Referring to the properties of software agents

presented in the previous sections, none of these are really existing in these implementations. The potential properties that are pretended to exist are autonomy, delegation and cooperation. However, delegation agents are not autonomous since management scripts must specify exactly what the agent should do. Agents cannot perform tasks without being ordered to do these tasks. They cannot decide on their own what, when, and how to do. These same critics apply to delegation: only low-level actions can be delegated. Finally, agents are not cooperative, in that they do not dynamically decide to share an important goal in order to achieve it in a coordinated manner. Nevertheless, the management by delegation concept is tacitly powerful and can be exploited in several ways. It can be the basis to support intelligent agents concepts within two potential manners. In a first way, one may conceive delegation agents as software agents instead. The second way consists in delegating intelligent agents as management scripts. This leads to remote-execution mobile agents. In both cases, further efforts, improvements and considerations must be addressed. Actually, an agent-oriented view of delegation will shift from the delegation of scripts and functions to the delegation of goals which is a high-level delegation.

Capitolo 4

Network Management with Mobile Agents

Mobile agents (MAs) are able to hop from node to node during their activities. Therefore, they can use the information they found during their past visit to sites in order to adjust their behaviour accordingly. They are also able to carry out activities close to where the needed resources are.

In [9], Thomas Magedanz has presented issues for the deployment of MAs for NM activities purposes. MAs can encapsulate management scripts and be dispatched on demand where needed. An agent can be sent to a network domain and travels among its elements collecting management data, and return back with the data filtered and processed. Sending a MA for this task is a substitute to performing low-level monitoring operations and process them centrally. Then if the agent is able to extract useful and concise information from raw data collected on each element, then the agent's size can remain small enough to save bandwidth usage. Actually, this is one of the most advanced arguments to promote MAs. Similarly, a MA can also be used to achieve management operations. Instead of remotely invoking management operations, the administrator can encapsulate these operations into a mobile agent and send it where needed. If these operations involve a lot of message exchange along with human decision making, then the mobile agent can also save bandwidth by locally taking the necessary decisions while performing those operations. These ideas have been implemented by FTP Software [10]. The IP Auditor is an application that dispatches mobile agents in the target network in order to collect manage-

ment information such as configurations and network elements' operation states. The IP Distributor is another application that dispatches agents that distribute management payload. Another original and concrete application of MAs is presented in [11]. Mobile agents are used to control traffic congestion in a circuit-switched network. A first class of MAs, called parent agents, randomly navigate among the network nodes and collect information about their utilization. By keeping track of these information, they have an approximated utilization average of the network nodes. Therefore, they are able to identify which nodes are congestioned relatively to this average. When a congestioned node is identified, a load-balancing MA is created. It has to update the routing tables of the neighboring nodes (using an optimal algorithm) so as to reduce the traffic routed by the congestioned node. When the network administrator can only be connected via an unreliable, costly or lossy link, he can create its MA offline, connect to the network to dispatch the agent, close the connection and then reconnect later to get his agent back with the results. In order to have a network flexibly managed, an administrator can connect to the network using its portable computer.

4.1 Recapping

The use of MAs seems to be rosy for NM Applications. However, some precautions must be taken before heavily investigating in this direction. Concerning bandwidth saving, it is not that carrying an agent with high capabilities consumes less bandwidth than carrying the desired information and processing it centrally. Actually, management information processing is often complex to achieve. Unless the agent is described in a high-level and concise language, it will be of a large volume. This leads to two options. Either the Agent runtime environment would be complex and heavy, or the agent has only limited capabilities. The first option leads to costly implementation of a MA platform while the second may simply be useless. Yet the design of a runtime environment for mobility which supplies the agents with a homogeneous interface across heterogeneous systems is not straightforward to conceive. The Java virtual machine is already a forward step but this is still far from the agent's concept.

Capitolo 5

Reactive, Deliberative and Hybrid Agents in Network Management

According to its architecture, an agent might be reactive, deliberative or hybrid [23]. While a reactive agent acts in situation-reaction or event-reaction manner, a deliberative agent acts according to decisions taken after a reasoning process over the environment. A hybrid agent is a combination of both.

5.1 Reactive Agents

The use of Reactive Agents [11] is particularly advantageous for the congestion management because MAs need to have a light weight not to overload the network by their mobility. The parent agent do not react unless they discover an overloaded node. The decision to react by creating a load agent is taken by simply comparing the currently visited node utilization with the average utilization computed during its traveling. The load agents which update the routing tables decide on the basis of an algorithm which doesn't imply any reasoning. Each agent taken alone, be it a parent or a load agent, would not be able to manage the network. The application is therefore a good example that shows how the overall multi-agent system goes beyond the capabilities of all the reactive agents when taken individually. Other researches concentrate on building agent-based frameworks for network ma-

nagement, and use the reactive architecture mainly for its simplicity. In [12], intelligent agents are applied to implement distributed management policies. Within this framework, agents are composed of a communication mechanism, a rule base, a solution base and an inference engine. The rule base contains rules that matches perceived events with the reactions to be taken. Reactions are also described by rules within the solution base. Therefore, the reactive rules are of the form event-Id,reaction-Id. This framework was used to achieve usage-based accounting. Another framework is described in [13]. It is defined and built using April++, an Object-Oriented extension of April [14]. The agent is organized in units which communicate (inside the agent) via a blackboard mechanism. The units are organized vertically which means that they execute concurrently. The local information of the agent is centralized in an information database. Besides, the agent has a link mediator which is an interface with the physical components of the network, and a head that insures the communication with other agents. New functions can be added to the agent by acquiring new units. This model has been applied to provide agents suitable for multi-service network management. Indeed, for this to be achieved, the agent is endowed with a customer unit to interact with the user, a service unit that honors service requests and a fault handling unit that deals with the faults that may occur.

5.2 Deliberative Agents

Two applications can be stated as deploying purely deliberative agents. The first is described in [15] and consists in an interesting framework applied to control a VPN service. Here agents are used to automate the negotiation that used to occur between the service provider and the service users when these ask to change the service parameters or to repair network faults. A customer agent has knowledge about the logical structure of the VPN and its usage, while the provider agent knows both the logical and physical implementation of each customer's VPN. Thus, agents have a model of the external world on which they act. Precisely, a customer agent knows for each trunk (or logical link) its capacity and its utilization, whereas the provider agent knows for each trunk the physical links on which it is implemented. When, let's say, a network fault occurs, the VPNs based on the faulty network element are affected. In this case, the customer agent first asks the

provider agent to repair the fault. In many cases, a complete and immediate repair is not possible and a negotiation-based cooperation is started. Customer agents try to find intermediate solutions based on their knowledge on the utilization of their respective trunks, and suggest partial solutions to the provider agent by updating the logical structure of their VPNs. The provider agent coordinates these solutions and makes the necessary updates in order to reach an accepted configuration. Both customer agents and provider agents use logical reasoning based on their models of the network and on their beliefs on its elements. Actually, the implementation of this reasoning is based on a planner called PRODIGY. The second application of deliberative agents in NM is the MANIA project (Managing Awareness in Networks with Intelligent Agents) [16]. An agent-based approach is used to manage the quality of service in the network. Agents are structured in beliefs, desires, intentions, goals and commitments. An agent's beliefs express its perception of the environment. A first part of the beliefs describe a real-time state of the network, e.g. the printing service is highly solicited at the current time. A second part contain the historical behavior of the network. The historical behavior is used to achieve some kind of learning about the dynamics of the network such as deducing that the printing service is highly solicited everyday from 10 to 11 am. A third part of the beliefs translate the states of the services provided over the network, e.g. the minimal response time the NFS server can ever have, or the maximum client number a web server can handle at a time. Finally, the agent may have beliefs on the user application contexts i.e user requirements in terms of QoS. The agent's desires consist of two parts. The first part of desires correspond to the requests that the agent could not satisfy, such as when a certain user requires a video connection while there is no available bandwidth (according to the agent's belief). The second part consists of motivation policies. The network administrator may want to "motivate" the agent to give a certain priority to some project members because they have a constraining deadline. When receiving a user-application context (which describes the user needed resources and the required quality of service parameters), the agent translates it into a set of goals. Goals are independent from the system's state. For example, a goal may state to "actively monitor the NFS server". These abstract goals are then mapped into intentions. Intentions take into account the current state of the system, the available means to achieve the goals and the possible constraints that may apply. Finally, and as the

administrator could motivate the agent, it is also able to specify obligation policies which form the commitment part of the agent.

5.3 Hybrid Agents

Here also two applications are worth presenting. The first is presented in [17]. It addresses the application of intelligent multi-agent systems to manage connection admission control in ATM. An ATM network is very dynamic and managing it centrally will not be appropriate because managed data get quickly out of date. But local management is not appropriate though because it lacks an overall view of the network. Here, the hybrid agents are particularly interesting. They may combine both a local real-time management via the reactive behavior, while the deliberative behavior is concerned with cooperating with other agents in order to achieve global planning and coordinate high-level tasks. In contrast, the second application is already well designed and the agent's architecture is well defined and is based on the concept of Vivid agents. Vivid agents [18] comprise both a reactive and a deliberative parts. However, the reactive part of the agent is not hardwired within the agent. Instead, the deliberative part may dynamically change the reactive behavior. Reagents are particular vivid agents with no planning capabilities. Reagents are applied for performing distributed diagnosis on distributed systems [19]. The network is partitioned in physical domains, each domain has its own diagnostic agent. The agent has a detailed knowledge model of its domain and minimal information about the neighboring domains, mainly the address of their respective agents. The multi-agent system applied a distributed version of the Model Based Diagnosis. The principle is that when an agent detects a fault, a lost connection for example, it starts by performing local diagnosis of its domain. This is the deliberative behavior of the agent, since it performs reasoning according to a model of its domain. If it finds no local fault, then it sends the resulting observations to the neighbor agent. The latter performs the same procedure. If it finds the faulty elements, it reports it to first agent. If the first agent itself had received the observation from another agent, then it must get it informed by forwarding the report. This is a reactive behavior of the agent where no reasoning is performed.

5.4 Recapping

Reactive agents are suitable for local and real-time management operations. They can be used efficiently in fixing well known faults within a relatively small portion of a network. An approach based on relating symptoms to faults, and faults to fixing methods is broadly adequate for this purpose. On the other hand, deliberative agents are much stronger from problem-solving and logical reasoning point of view than reactive agents. This makes them require much more resources, and makes their response time rather long. For this reason, they are suitable for complex but not time-constrained tasks. A good example would be that of analyzing the origin of a security failure in a large corporate network. Further, the idea of using hybrid agent to take the advantages of both architectures is very interesting. The concept of vivid agents is a good basis to implement this idea. Vivid agents offer an increased flexibility when designing an agent since their reactive behavior could be updated during run-time. This could be applied for example to support complex learning algorithms where the agent, still having timely reactions, has its behavior improved in time. However, one may keep in mind that the same benefits could be obtained using a heterogeneous architecture, i.e. a multi agent system with both reactive and deliberative agents with the latter governing the behavior of the former.

Capitolo 6

Cooperation in Multi-Agent Systems for Network Management

Cooperation is one of the most important properties in MASs [20]. In NM, cooperation could have large benefits in Distributed Network Management Systems. Though the term "cooperation" (or "collaboration") is unclearly used, i will try in the following subsections to describe cooperation in agent-based network management approaches progressively starting from no or simple coordination towards more elaborated cooperation.

6.1 Coordination without Communication

The Multi Agent Systems described previously are based on agents which do not directly communicate. An agent may be aware of the other agents only by observing the system's state. Each time a parent agent visits a node, it leaves therein some information such as its identity, its age and the date of its visit. By reading this information in every visited node, the parent agents are aware of their number in the network. If their number is excessive, then the youngest parent agent will terminate. Furthermore, in determined nodes, static processes continuously check if all the parent agents are still alive and did not crash. If a very long period has elapsed from the last visit of a certain parent agent, then the static process will conclude that this parent has crashed and will replace it. Besides, when a

load agent is launched on a node, it creates a record with its identity and its start time. When it finishes updating the routing tables, it returns back to the first node and registers its same start time again and then terminates. Therefore, any parent agent that visits that node is able to know if the load agent is still active or not. It is also able to detect that the load agent has crashed if the start time has not been written for the second time after a long period. The parent agent becomes responsible of creating another load agent to replace the first. When there is no direct communication between agents, a lot of coherence problems are avoided. Firstly, an agent would not send messages to another agent that has been crashed. Besides, all the agents equally share the same information and no mechanisms are needed to preserve coherence. Secondly, the agents are able to coordinate their tasks in a simple way. These properties gave the Multi Agent System thus designed a high degree of robustness and graceful degradation which are rare (actually never) to be considered in the other applications.

6.2 A primitive cooperation

In section 5 it was presented an agent-based system implementing a distributed version of Model-Base Diagnosis [21] in case of a fault, an agent performs local diagnosis and if it finds no faulty element, it asks the next agent to perform diagnosis in its corresponding domain. One may consider this as a simple cooperation for at least two reasons. The first reason is that the underlying communication mechanism is itself primitive. The second is that this cooperation is hard coded, and the agents have not decided how to carry it out. However, referring to [22], agents are cooperating since they participate to achieve a global goal which is to identify the faulty element in a large network.

6.3 Self-interested Negotiation Agents

Self-interested agents do not need to know about the other agents' plans and goals. Instead, they work in order to achieve their own goals in the best way without caring about the others. However, in order to achieve a goal, a self-interested agent may need the services of some other agents. A natural approach would be based upon negotiation. Here, when requiring some task

to be done by another, the agent tries to maximize its profit by launching a kind of auction. Self-interested negotiation agents seem to be very suitable for some network management activities, especially for service management. Most of the papers dealing with service management refer to this kind of agents. For example some agents are adequate for the provision of high-level services in an open electronic market of telecommunications services. A good basis to cover this trend is the FIPA application. It defines three kinds of agents, namely the Personal Agent (PA), the Service Provider Agent (SPA), and the Network Provider Agent (NPA). The latter is responsible for the provision of the network resources and elements which are necessary for the service implementation. Based on these network resources, the SPA is responsible for the provision of the services with the expected quality. The PA is a kind of Personal Digital Assistant that assists the user in defining his requirements for the application's needs with regard to the user's preferences. It then has to negotiate these requirements with different SPAs in order to find out the best provider with the best service in terms of maximum quality and minimum cost. When a communication is about to be established, the PA has to commit the local resources of the user and to configure his equipment according to the established connection. The SPA has to catch the user's requirements and to identify the necessary services and to map them into these services' parameters. It then negotiate with the NPAs to select the best network provision, again in terms of maximum quality with minimum cost. Feedback with the PA may occur in order to conclude with the best arrangement both with the Network Provider and the User. Finally, the NPA gets the SPA's specifications and translate them into network requirements in terms of bandwidth, jitter, etc. It may also have to negotiate with other NPAs, mostly in the case where multiple network domains are implied. Other works on the subject, by Mike Ruzzo and Ian A. Utting [24], introduce User Agents to allow for the definition of customized services. New services can be customized by supplying the agent with the user's policy. The UA then is able to make the best choice between what can be achieved as suggested by the fall-back of the called agent, and what the service provider offers. The caller and the called agents may also negotiate in order to reach an agreement that satisfies both users' policies i.e. preferences and constraints. To support user mobility, end-point agents are also introduced. Mobile-user agents have to negotiate with end-user agents to get the permission to access the corresponding end-points and

establish communications.

6.4 Delegation-based Cooperation

As opposed to the MbD paradigm which only deals with script delegation, we address here the delegation between intelligent agents. This is a higher-level delegation because agents delegate goals and motivations instead of low-level operations. The agent framework described in [12], makes use of policies expressed in logical rules. High-level policies are delegated to agents. These policies are then mapped into lower-level policy rules and stored in the agent's rule base. The MANIA project also makes use of high-level delegation of goals. An agent may fail in achieving a goal mainly because it lacks the necessary resources. It, therefore, delegates that goal to another agent. What is important to note is that only the goal and not the associated intentions is delegated. The latter agent rebuilds the entire intentions for that goal and these intentions may differ from those built by the delegating agent due to different beliefs, commitments and skills of the two agents.

Capitolo 7

A General Review - Conclusions

7.1 An agent-oriented view

- Pro-activeness: pro-activeness is the ability for an agent to anticipate changes in the environment and act in a way to prevent the potential forthcoming problems that might occur. In the NM context, pro-activeness makes an agent able to foresee faults or QoS performance degradation and to avoid such situations before they even occur. Pro-activeness is the most ignored property in existent agent-based approaches to NM.

- Learning:Learning Network Management knowledge or expertise should be distinguished from learning the network's behavior. The first kind of learning [25] enables the agent to acquire new capabilities in management, mainly to handle new types of network problems. The second kind of learning [11] enables the agent to acquire knowledge on the network it manages in order to apply its very same expertise but more efficiently.

- Robustness and Graceful Degradation: these two properties concern the whole multi-agent system. The graceful degradation means that the MAS does not fail drastically at the boundaries. Robustness is the ability to recover from faults that may occur due to the underlying environment (network failure or system halt). The mechanism that

detects crashed agents and replace them was already presented in the previous sections. Let's just recall that if a parent agent crashes, it will be detected and replaced by static processes whose role is to maintain a minimum number of agents. Meanwhile, the other agents continue their tasks without stopping the system (graceful degradation). If a load agent crashes, it will be detected and replaced by the parent agents. Therefore, the MAS keeps on performing its role and is able to recover from possible crashes (robustness).

7.2 Network Management trends

The number of network management firms and network-device manufacturers which are getting interested in Intelligent Agent technology is rapidly increasing. Many manufacturers are trying to incorporate IAs within network devices such as routers and switches. The idea behind is to endow the network devices with a level of intelligence that allows them to acquire the properties of autonomy and self-healing. Besides, this intelligence at the lowest level of the network will have benefits on management traffic reduction and network management automation. Other trends are focusing on the service level. These trends are mostly lead by telecommunications operators which are addled for a high degree of flexibility in establishing new services and for means to control and satisfy the quality of service that the end user requires. Network management platforms are also affected by IAs. Many manufacturers are announcing products supporting IAs, although it is not yet specified how IAs will be integrated in their platforms. For example, the whole NM platform can be based on IAs, or it can include a framework to allow network administrators to develop their own NM intelligent agents, or simply can use static preprogrammed intelligent agents. Their hope is for the NMS to be much more reactive (or even pro-active), flexible and scalable. In fact, a lot of researches are still needed to reach an acceptable usage of IA technology in NM platforms. Indeed, a huge number of languages, environments and techniques are available to design agents and to define their behavior and how they can cooperate. Besides, there is still a lot of yearning in the NM field in order to find solutions to integrate heterogeneous and vendor-specific equipments. Regardless of these issues, efforts are directed towards Java-based solutions. People are relying on the promise of

a portable code due to the Java virtual machine and to Java-enabled chips which are announced and even recently commercialized.

Bibliografia

[1] Aiko Pras. *Network Management Architecture*. PhD thesis, Universitet Twente, Department of Computer Science, 1995.

[2] Douglas W. Stevenson. *Network management: What it is, what it isn't*, 1995.

[3] Stan Franklin and Art Graesser. *Is it an agent, or just a program?: A taxonomy for autonomous agents.* Agent Theories, Architectures, and Languages. Springer, 1996.

[4] Michael Wooldridge and Nicholas R. Jennings. *Intelligent Agents: Theory and Practice*, 1995.

[5] Jorg P. Muller. *The Design of Intelligent Agents - A Layered Approach. LNAI State-of-the-Art Survey.* Springer, Berlin, Germany, 1996.

[6] Yechiam Yemini, Gerrnan Goldszmidt, and Shaula Yemini. *Network Management by Delegation.* April 1991.

[7] Gerrnan Goldszmidt. *Distributed system management via elastic servers.* 1993.

[8] Gerrnan Goldszmidt and Yechiam Yemini. *Distributed Management by Delegation.*1995.

[9] Thomas Magedanz. On the impacts of intelligent agents concepts on future telecommunication environments, in *Third International Conference on Intelligence in Broadband Services and Networks*, 1995

[10] FTP Software. Ftp software agent technology. Technical report, FTP Software, http://www.ftp.com/product/whitepapers/4agent.htm, 1997

[11] S. Appleby and S. Steward. *Modelling Future Telecommunication Systems*, Chapman Hall, 1994.

[12] P. Steenekamp and J. Roos. *Implementation of distributed systems management policies: A framework for the application of intelligent agent technology*, 1996.

[13] Nikolaos Skarmaeas and Keith L. Clark. *Process oriented programming for agent-based net- work management*, 1996

[14]] F. G. McCabe and K. L. Clark. *April: Agent process interaction language.* 1994.

[15] Robert Weihmayer and Ming Tan. *Modelling cooperative agents for customer network control using planning and agent-oriented programming*, 1992.

[16] Raul Oliveira and Jacques Labetoulle. *Intelligent agents : a way to reduce the gap between applications and networks.*

[17] M. A. Gibney and N. R. Jennings. *Market based multi-agent systems for atrn network management*, 1997

[18] Gerd Wagner. *Vivid agents - how they deliberate, how they react, how they are verified*, 1996

[19] Peter Frohlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schroeder. *Diagnostic agents for distributed systems*, 1997

[20] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. *Multi-agent systems*, 1997

[21] Peter Frohlich, Iara de Almeida Mora, Wolfgang Nejdl, and Michael Schroeder. *Diagnostic agents for distributed systems* Sienna, Italy, Junary 1997.

[22] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. *On cooperation in multi-agent systems. The Knowledge Engineering Review*, 1997

[23] K. Terplan. *Communication Networks Management.* Prentice-Hall, 1992.

[24] Mike Rizzo and Ian A. Utting. *An agent-based model for the provision of advanced telecommunications services*, 1995.

[25] Babak Esfandiari, Gilles Deflandre, and Joel Quinqueton. *An interface agent for network supervision*, 1996.